Part2

Do You Know?
Set 2

The source code for the BoxBug class can be found in the boxBug directory。.

1. What is the role of the instance variable sideLength?
The role of the instance variable sideLength is the number of the steps that a boxBug can move in each box side.

2. What is the role of the instance variable steps?
The role of the instance variable steps keep the number of steps that a boxBug has moved and track the current steps.

3. Why is the turn method called twice when steps becomes equal to sideLength?
Because the bug has to turn to a new direction, each turn only turning 45 degrees. Calling twice turn method can turn 90 degrees.

4. Why can the move method be called in the BoxBug class when there is no move method in the BoxBug code?
Because the BoxBug class extends the Bug class, and the move method is define in the Bug class. Box class the super class of the BoxBug, so that BoxBug inherites move method from Box class.

5. After a BoxBug is constructed, will the size of its square pattern always be the same? Why or why not?
Yes, after the size of its square has been determined when a BoxBug is constructed, it can not be changed.

6. Can the path a BoxBug travels ever change? Why or why not?
Yes, If a actor, rock or a bug is in the front of the BoxBug, the BoxBug would change to a new path without changing the sizeLength.

7. When will the value of steps be zero?
When the BoxBug is constructed, the value of the steps is initialized to zero. When the the value of is equal to the value of sideLength, the value of steps be zero.
The BoxBug reach the size of box turning to a new side.

Exercises

In the following exercises, write a new class that extends the *Bug* class. Override the *act* method to define the new behavior.

1. Write a class CircleBug that is identical to BoxBug, except that in the act method the turn method is called once instead of twice. How is its behavior different from a BoxBug?

The path of the boxbug is a square, but the circlebug's is a octagon.

2. Write a class SpiralBug that drops flowers in a spiral pattern. Hint: Imitate BoxBug, but adjust the side length when the bug turns. You may want to change the world to an UnboundedGrid to see the spiral pattern more clearly.

See the code.

3. Write a class *ZBug* to implement bugs that move in a "Z" pattern, starting in the top left corner. After completing one "Z" pattern, a *ZBug* should stop moving. In any step, if a *ZBug* can't move and is still attempting to complete its "Z" pattern, the *ZBug* does not move and should not turn to start a new side. Supply the length of the "Z" as a parameter in the constructor. The following image shows a "Z" pattern of length 4. Hint: Notice that a *ZBug* needs to be facing east before beginning its "Z" pattern.
See the code.

4.Write a class *DancingBug* that "dances" by making different turns before each move. The *DancingBug* constructor has an integer array as parameter. The integer entries in the array represent how many times the bug turns before it moves. For example, an array entry of 5 represents a turn of 225 degrees (recall one turn is 45 degrees). When a dancing bug acts, it should turn the number of times given by the current array entry, then act like a Bug. In the next move, it should use the next entry in the array. After carrying out the last turn in the array, it should start again with the initial array value so that the dancing bug continually repeats the same turning pattern.The DancingBugRunner class should create an array and pass it as aparameter to the DancingBug constructor.
See the code.

5. Study the code for the BoxBugRunner class. Summarize the steps you would use to add another BoxBug actor to the grid
a. first step is to create a BoxBug object using the BoxBug constructor with the desire sideLength.
b. second step is to add the BoxBug object to the gridWorld using the world.add() method with a desire
valid location.